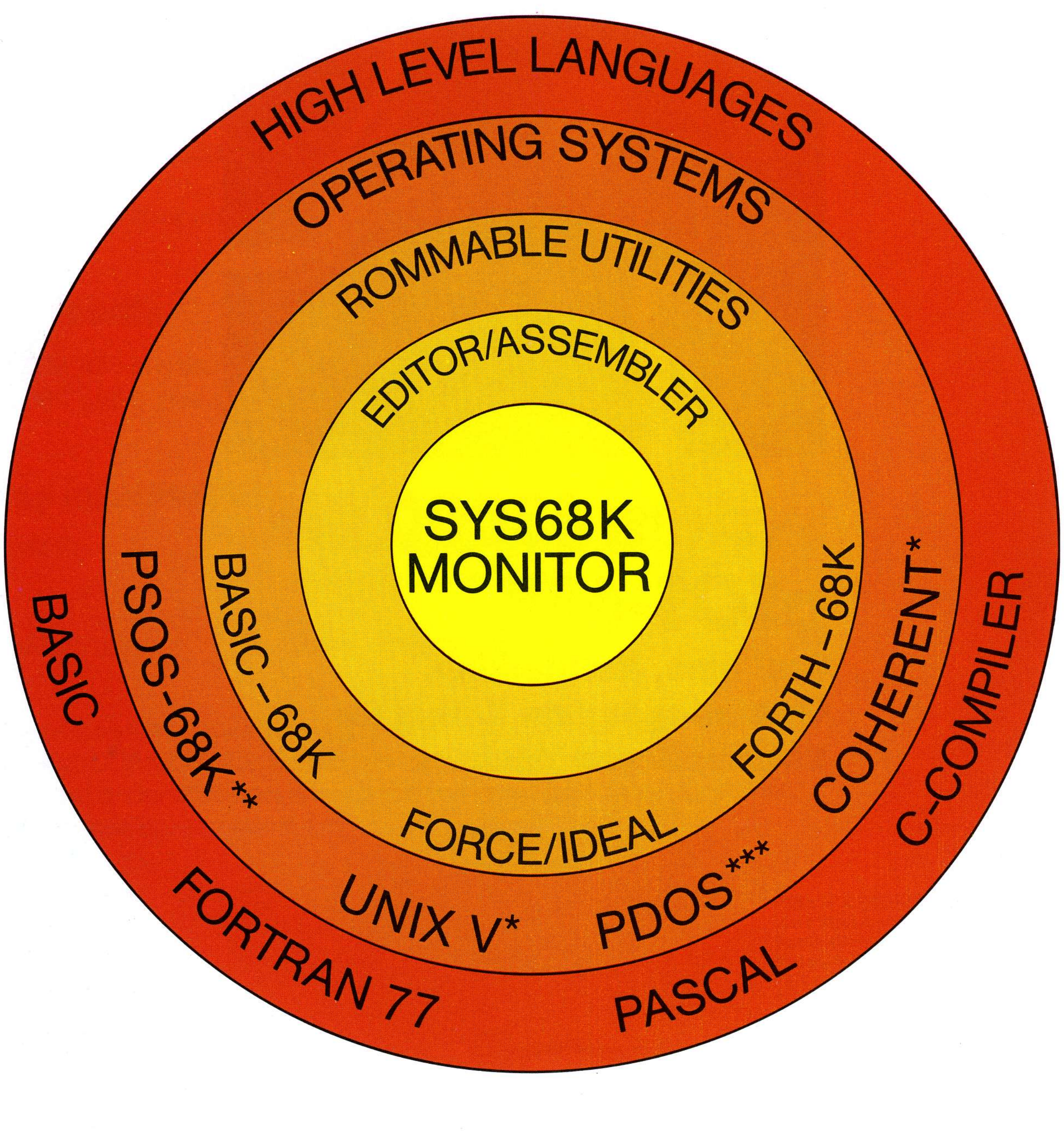
System 68000 VME

pSOS-68K

Real time, Multitasking, Operating System Kernel for VMEbus Board Modules

SYSTEM FIRMWARE & SOFTWARE



General Description

The pSOS-68K package is an implementation of the pSOS* software environment on board level products of the VME family of FORCE COMPUTERS etc. In its customized form it represents a new direction in software components comparable to that achieved with LSI technology in hardware designs. pSOS is shipped in EPROM and can be used like a floating point coprocessor using the standard pSOS-68K calling sequences.

Features of the EPROM resident pSOS-68K package:

- Plug in, silicon operating system
- Requires no system generation, compilation, or linking
- Usable with any development system
- Totally hardware independent
- Self configuring and position independent
- Easy to use interface
- Device independent logical I/O interface
- User and privileged processes

pSOS-68K is a real time, multitasking operating system nucleus for the 68000 CPU. It is excellent as a fast multitasking executive in real time, ROM based, SBC applications. Moreover pSOS-68K can serve as the OS Kernel and I/O supervisor in memory mapped, disk based real time systems.

*) pSOS-68K is a trademark of Software Components Group, Santa Clara / California

Functional Overview

pSOS-68K constitutes a layer of supervisory software around the 68000 CPU. Its rich feature set and simple interface encourage an efficient, structured application design, and improve productivity through reduced design and debugging efforts.

The user application is partitioned into logically concurrent processes. Related processes can be grouped, forming protected subbranches in the process tree. pSOS schedules and allocates resources between processes, and supervises asynchronous processing. Processes are scheduled by priority; equal priority processes are roundrobined by time-slice.

User processes and I/O drives interface with pSOS-68K via a set of extended 68000 instructions which comprise the kernel service calls and I/O supervisor calls.

Kernel Service Requests

The twenty-nine kernel calls serve five groups of functions.

Process Management
The first group of service calls allow dynamic creation and deletion, and control of processes.

Memory Management
These calls perform dynamic memory allocation and reclamation, and control the mapping and protection hardware, if available. pSOS provides an efficient »buddy« allocation algorithm for managing unmapped memory. This serice group is designed to be replaceable to work with any MMU, including the Motorola 68451 and the Stanford SUN segmentation-paging scheme. All MMU-related changes are localized to these nine calls; no other changes to pSOS are required or recommended.

Interprocess Communication
These primitives allow processes to communicate, synchronize, and effect mutual exclusion, and thus perform crucial services in a multitasking environment. The Message Buffer facilities provided are efficient and multi-purpose. Request/send_messages, for example, supplant P/V semaphore primitives and, moreover, allow recla-

mation of resource-related messages upon asynchronous process deletions. Messages are routed via exchanges which provide effective N-process to M-process communications.

Time-Base Management

If RTC hardware is included, time-base services provide a real-time clock, allow processes to pause for specified periods, and provide time-out options to other kernel services such as Request_message and Allocate_memory calls.

Miscellaneous

The RETI and DISPATCH calls allow Interrupt Service routines and Privileged processes respectively to invoke a pSOS dispatch cycle.

I/O Supervisor Requests

The pSOS I/O supervisor imposes a logical, device-independent structure on user-provided device drivers. I/O requests are made via standard calls to the I/O Supervisor, employing logical device numbers. Each device driver must present six procedures, some of which may be simply null or error returns. I/O operations are designed for efficiency, and require no lengthy request blocks. Moreover, highly time-critical I/O can be performed outside the I/O Supervisor, using Interprocess Communication services between the interrupt service routines and the servicing processes.

TABLE I – Service and I/O calls

Spawn_process
Activate_process
Delete_process
Identify_process
Suspend_process
Resume_process
Get_process_priority
Allocate_segment
Free_segment
Assign_segment
Attach_section*
Detach_section*
Lock_section*
Unlock_section*
Load_process_to_MMU*
Log_to_phys_address_xiate*
Create_exchange
Delete_exchange
Attach_exchange
Send_message
Request_message
Signal_process.event
Wait_event
Announce_tick
Pause
Set_time_of_day
Get_time_of_day
Ret_from_interrupt
Dispatch
Device_init
Device_open
Device_close
Device_read
Device_write
Device_control

*For MMU implementation only

Building an Application

pSOS-68K is uniquely versatile and easy to use. Figure 1 depicts the components in a typical application, where dashed lines suggest data connectivity, and solid arrows indicate action flow paths.
User software consists of two types of objects – Processes and I/O drivers. A small Configuraton Table supplies all hardware and application-dependent parameters to pSOS, including address and size of free RAM, RTC frequency, number and location of I/O drivers, and the user Rootprocess descriptor.
After a system reset, the following modules must be memory resident-pSOS, Configuration Table, Root_process, and Device drivers. Some or all of these modules may be ROM'ed or bootloaded. Similarly, the Vector page may be ROM'ed or initialized by a small Bootprom monitor to point to the kernel entries and user interrupt handlers. The reset procedure then branches to the pSOS initializer with a pointer to Configuration Table. Per the Configuration parameters, the initializer carves a kernel data segment from the free RAM area, and sets up its data and list structures. The Init_procedure in each Device driver is then called in turn, which resets the Device and initializes its data area for itself, if necessary. The kernel initializer then calls a routine which sizes up the remaining unused RAM and sets up the Memory Manager.
The last initialization step actives the user Rootprocess, whose responsibility it is to spawn and activate all user processes needed at startup time. Thereafter, processes communicate with each other, spawn offspring processes, share resources, and perform I/O, all through pSOS.

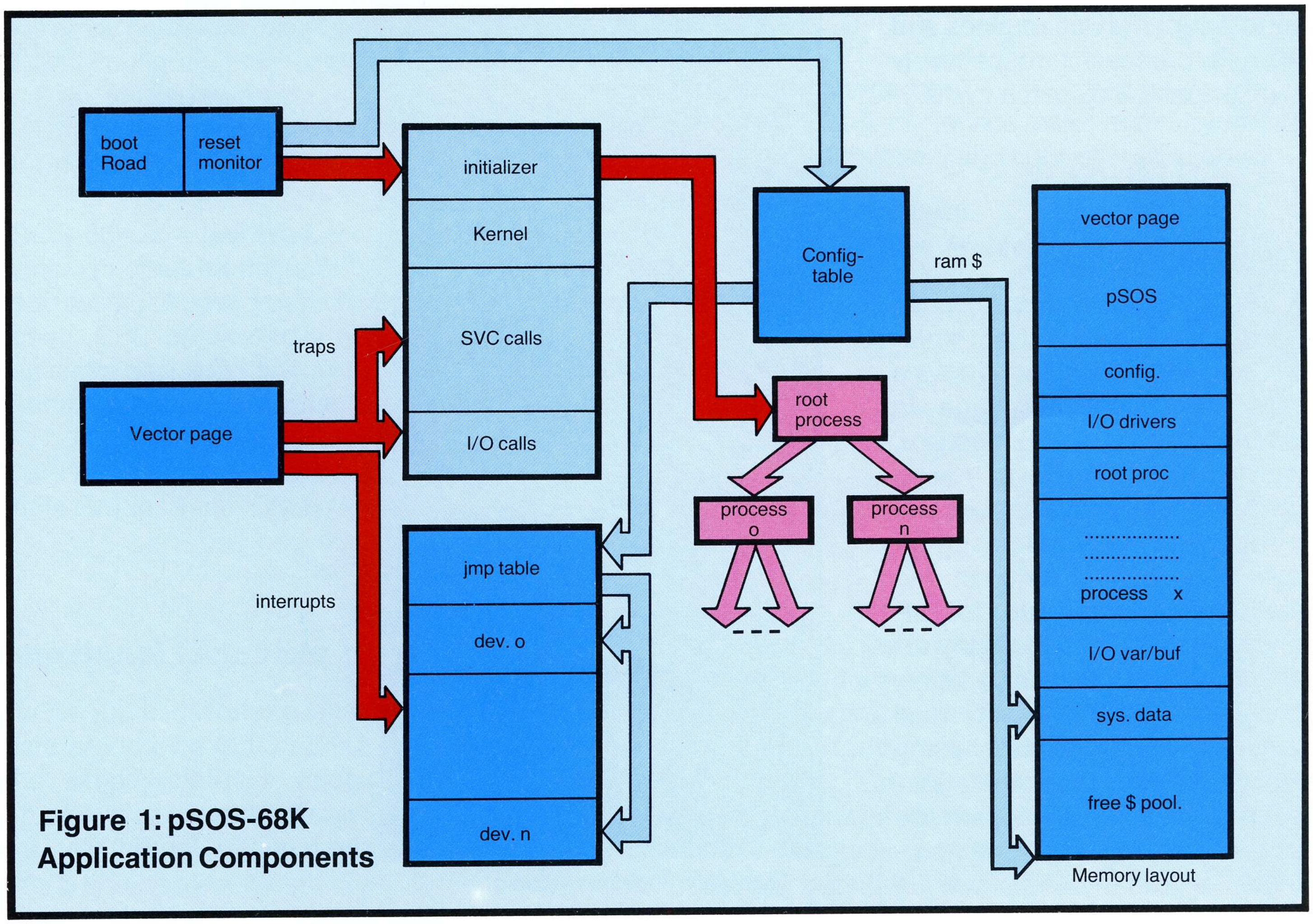


Figure 1: pSOS-68K Application Components

Specifications

Hardware Requirements:

- 68000 CPU
- Memory 4KB code, 512 bytes data minimum
- RTC Real time clock - recommended

Standard Features:

- Kernel with Memory Manager (unmapped)
- I/O supervisor
- Exception logger/reporter (virtual console)

Shipping Media

- 2764 EPROM set containing pSOS firmware
- pSOS-68K users' documentation

Ordering Information

SYS68K/pSOS Part No. 130001	EPROM chip set including documentation, single evaluation license
SYS68K/pSOS/UM Part No. 800018	Users' documentation

Note: The pSOS-68K package is copyrighted and licensed by FORCE COMPUTERS GmbH and may only be used in accordance with and under the terms and conditions of such a license agreement (see license agreement form).

FORCE COMPUTERS INC.
2041 Mission College Blvd.
Santa Clara, California 95054
Phone (408) 988-8686
TLX 172465
FAX (408) 9809331

FORCE COMPUTERS GmbH
Daimlerstraße 9
D-8012 Ottobrunn
Telefon (0 89) 6 09 20 33
Telex 5 24 190 forc-d
Telefax: (0 89) 6 09 77 93

FORCE COMPUTERS FRANCE Sarl
11, Rue Casteja
92100 Boulogne
France
Phone (1) 620 37 37
TLX 2 06 304 forc-f

1284/05.0/pSOS/A2 Printed in West Germany by DMR